Politechnika Wrocławska

Wydział Matematyki
Kierunek studiów: Matematyka stosowana
Specjalność: –

Praca dyplomowa – inżynierska

ZASTOSOWANIE STOCHASTYCZNEJ OPTYMALIZACJI DO GIER CZĘŚCIOWO OBSERWOWALNYCH

Aleksander Jakóbczyk

słowa kluczowe:
Gry częściowo obserwowalne, strategia optymalna, stochastyczna optymalizacja, Hoeffding race, empirical Bernstein race, algorytmy genetyczne.

krótkie streszczenie:
Celem pracy jest wyznaczenie nieoczekiwanych strategii w grach częściowo obserwowalnych za pomocą metod stochastycznej optymalizacji. W prawy zostały wprowadzone nowe algorytmy pozwalające na znacznie szybsze wyznaczenie lepszej strategii niż ich oryginalne odpowiedniki. Praca opiera się na wynikach Cauwet i Teytauda z 2018 roku, w których przedstawiono nieoczekiwane strategie dla kilku klasycznych gier oraz kilku metod optymalizacji. Przeprowadzana również została analiza porównawcza dla kilku różnych metod optymalizacji.

Opiekun pracy dyplomowej	dr inż. Andrzej Giniewicz	.....	.....
	Tytuł/stopień naukowy/imię i nazwisko	ocena	podpis

*Do celów archiwalnych pracę dyplomową zakwalifikowano do:**

a) kategorii A (akta wieczyste)

b) kategorii BE 50 (po 50 latach podlegające ekspertyzie)

** niepotrzebne skreślić*

pieczątka wydziałowa

Wrocław, rok 2023

Wrocław University
of Science and Technology

Faculty of Pure and Applied Mathematics
Field of study: Mathematics
Specialty: Theoretical Mathematics

Engineering Thesis

APPLICATION OF STOCHASTIC OPTIMIZATION TO PARTIALLY OBSERVABLE GAMES"

Aleksander Jakóbczyk

keywords:
Partially observable game, optimal strategy, stochastic optimization, Hoeffding race, empirical Bernstein race, genetic algorithm

short summary:
The aim of this work is to determine unexpected strategies in partially observable games using stochastic optimization methods. New algorithms have been introduced that allow for much faster determination of a better strategy than their original counterparts. This work is based on the results of Cauwet and Teytaud from 2018, in which they presented unexpected strategies for several classical games and several optimization methods. Additionally, a comparative analysis of several optimization methods is conducted.

Supervisor	dr inż. Andrzej Giniewicz	.....	.....
	Title/degree/name and surname	grade	signature

*For the purposes of archival thesis qualified to:**

a) category A (perpetual files)

b) category BE 50 (subject to expertise after 50 years)

** delete as appropriate*

stamp of the faculty

Wrocław, 2023

Spis treści

Wstęp

Gry dwuosobowe, w których gracze posiadają niepełną wiedzę o swoich ruchach i możliwych decyzjach przeciwnika, stanowią istotną kategorię problemów w matematycznej teori gier. Uzyskiwane roziwazania w rozawzanej tetori mają zastosowanie w rzeczywistym życiu. W niniejszej pracy skupimy się na probabilistycznych metodach znajdowania lepszej strategii oraz zastosujemy te metody do wyznaczenia optymalnej strategii (w sensie Nasha [10]) w rzeczywistych grach dwuosobowych częściowo obserwowanych. Praca składa się z sześciu rozdziałów. W pierwszym z nich zostanie przedstawiona podstawowa wiedza dotycząca problemu określania strategii optymalnej. W tym rozdziale zostaną również wprowadzone podstawowe twierdzenia, na których opiera się działanie analizowanych algorytmów oraz zostanie przedstawiony problem porównań wielokrotnych. W następnym rozdziale przedstawimy i omówimy algorytmy racingowe, których celem jest określenie, która z porównywanych strategii jest lepsza. Wprowadzimy również pewne modyfikacje do bazowego algorytmu oraz przeprowadzimy analizę prawdopodobieństwa popełnienia błędu dla nowo zaproponowanych metod. W rozdziale trzecim omówimy działanie algorytmów służących do wyznaczania optymalnej strategii. Czwarty rozdział będzie poświęcony opisowi gier, dla których poszukamy optymalnych strategii. Ostatnim rozdziałem będzie prezentacja wyników działania algorytmów genetycznych oraz przeanalizujemy strategii, które udało się za ich pomocą odkryć.

Rozdział 1

Definicje, lematy, twierdzenia, przykłady i wnioski

Celem algorytmów, które będziemy wykorzystywać, jest znalezienie optymalnej strategii dla graczy w dwuosobowych grach częściowo obserwowalnych. Aby matematycznie opisać pojęcia gry i strategii optymalnej, wprowadzimy kilka podstawowych definicji. Definicje dotyczące podstaw teorii gier pochodzą z prac [8], [9].

1.1 Czym jest gra?

Omówmy zatem, co nazywamy grą w matematyce.

Gra w matematycznej teorii gier to sytuacja, w której gracze wybierają swoje strategie i otrzymują nagrody lub kary w zależności od podjętych decyzji oraz losowych zdarzeń. Teoria gier pozwala na modelowanie i analizowanie takich sytuacji oraz umożliwia znajdowanie rozwiązań, które są optymalne dla graczy.

Matematyczna definicja gry w teorii gier zazwyczaj zaczyna się od określenia następujących elementów:

- Zbioru graczy: Jest to zbiór wszystkich osób biorących udział w grze. Zbiór ten składa się z co najmniej dwóch elementów w zależności od typu gry.
- Zbioru strategii: Jest to zbiór wszystkich możliwych strategii, które mogą być wybierane przez graczy. Strategia to plan działania gracza, który zakłada, jakie ruchy gracz zamierza podjąć w danej grze.
- Funkcji zwrotu: Funkcja zwrotu określa nagrody lub kary, które gracze otrzymują w zależności od ich strategii i losowych zdarzeń.
- Struktury informacji: Struktura informacji określa, jakie informacje są dostępne dla graczy w momencie podejmowania decyzji. Informacje mogą być pełne lub częściowe, co wpływa na możliwe strategie graczy i ich skuteczność.

Gry mogą być podzielone na kilka kategorii w zależności od kilku różnych kryteriów. Jednym z takich kryteriów jest moment, w którym gracze podejmują decyzje:

Definicja 1.1 (Gra w postaci strategicznej)**.** Jest to typ gry, w której gracze podejmują decyzje w tym samym momencie.

Definicja 1.2 (Gra w postaci ekstensywnej)**.** Jest to typ gry, w której gracze podejmują decyzje we wcześniej ustalonej kolejności.

Przykładami gier w postaci strategicznej są gry kamień papier nożyce, oszust czy też mora. Natomiast przykładami gier w postaci ekstensywnej są szachy, warcaby oraz go.

Gry możemy również dzielić ze względu na posiadaną wiedzę.

Definicja 1.3 (Gra z kompletną informacją)**.** Jest to typ gry, w której gracze mają informacje o możliwych przyszłych wynikach gry i o zbiorach możliwych strategii.

Definicja 1.4 (Gra częściowo obserwowalna)**.** Jest to przeciwieństwo gier z kompletną informacją.

Przykładami gier z kompletną informacją są szachy, warcaby oraz go. Natomiast przykładami gier częściowo obserwowalnych są wszelkie gry posiadające w rozgrywce pewne elementy losowe takie jak rzut kostką czy też dobieranie kart.

Istnieje jeszcze wiele innych podziałów gier ze względu na kategorie takie jak liczba graczy, zbiory dostępnych akcji, możliwość tworzenia koalicji i wiele innych.

1.2 Definicje i Oznaczenia

1.2.1 Strategie proste

Wprowadźmy podstawowe oznaczenia potrzebne nam do tego, aby móc zdefiniować czym jest strategia optymalna:

- $N = \{1, 2, \ldots, n\}$: zbiór graczy,
- $A_i, i \in N$: niepusty zbiór strategii czystych gracza i,
- $m_i = |A_i|$: liczba strategii gracza i,
- $A = \prod_{i \in N} A_i$: zbiór wszystkich strategii gry,
- $u_i : A \to \mathbb{R}$: funkcja wypłaty gracza i,
- $a = (a_1, a_2, \ldots, a_n) = (a_i)_{i \in N}$, $a_i \in A_i$: profil gry w strategiach czystych,
- $u_i(a) = u_i(a, a_{-i})$: wypłata gracza i z profilu a,
- $a_{-i} = (a_i)_{i \in N \setminus \{i\}}$: profil wszystkich strategii poza strategią graca i.

Definicja 1.5 (Gra strategiczna)**.** Grą strategiczną nazywamy trójkę $GS = \langle N, (A_i)_{i \in N}, (u_i)_{i \in N} \rangle$.

Definicja 1.6 (Równowaga Nasha w strategiach czystych gry strategicznej)**.** Równowaga Nasha w strategiach czystych gry strategicznej jest to profil gry $a^* = (a_1^*, a_2^*, \ldots, a_N^*) \in A$, taki, że

$$\forall\, i \in N \ \forall\, a_i \in A_i \quad u_i(a_i^*, a_{-i}^*) \geqslant u_i(a_i, a_{-i}^*).$$

Zatem jest to profil gry, w którym istnieje strategia czysta, dająca nie gorsze wyniki od dowolnej innej strategii czystej. Okazuje się jednak, że taki stan nie zawsze istnieje w strategiach czystych, np. w grze kamień papier nożyce strategia grania tylko kamienia daje gorszy rezultat przeciwko strategii grania tylko papieru. Podobnie ze strategią grania tylko nożyc i grania tylko papieru.

1.2.2 Strategie mieszane

Definicja 1.7 (Strategia mieszana)**.** Strategią mieszana σ_i graca i w grze strategicznej $GS = \langle N, (A_i)_{i\in N}, (u_i)_{i\in N}\rangle$ nazywamy rozkład prawdopodobieństwa na zbiorze strategii czystych A_i

$$\sigma_i = (\sigma_{i1}, \sigma_{i2}, \ldots, \sigma_{im_i})$$

gdzie σ_{ik} oznacza prawdopodobieństwo, że gracz i zagra strategię czystą $k \in A_i$.

Fakt 1.8. *Strategia czysta jest szczególnym przypadkiem strategii mieszanej, w którym prawdopodobieństwo zagrania jednej z dostępnych strategii wynosi 1.*

Wprowadźmy dodatkowe oznaczenia:

- $\Sigma_i = \{\sigma_i : A_i \to [0,1], \sum_{k=1}^{n} \sigma_{ik} = 1, \sigma_{ki} \geqslant 0\}$: Zbiór strategii mieszanych gracza i,
- $\sigma = (\sigma_1, \sigma_2, \ldots, \sigma_n)$: Profil gry,
- $u_i(\sigma) = u_i(\sigma_i, \sigma_{-i})$: Wypłata gracza i z profilu σ,
- $\sigma_{-i} = (\sigma_i)_{i\in N\setminus\{i\}}$: Profil wszystkich strategii poza strategią gracza i.

Definicja 1.9 (Równowaga Nasha w strategiach mieszanej gry strategicznej)**.** Profil gry strategicznej σ_i^* jest Równowagą Nasha, gdy

$$\forall\, i \in N \;\; \forall\, \sigma_i \in \Sigma_i \quad u_i(\sigma_i^*, \sigma_{-i}^*) \geqslant u_i(\sigma_i, \sigma_{-i}^*).$$

Równowagę Nasha interpretujemy jako taki profil gry, w którym żadnemu z graczy nie opłaca się zmieniać swojej strategi, ponieważ nie skutkuje to zwiększeniem swoich zysków.

1.3 Twierdzenia

Algorytmy 1, 2, 3, 4, 5 wykorzystywane w poniższej pracy oparte są o dwa twierdzenia, a dokładniej o szczególne przypadki wynikające z nierówności 1.10 i 1.12 [5].

Twierdzenie 1.10 (Nierówność Hoeffdinga)**.** *Niech $X_1, X_2, \ldots, X_t$ będzie ciągiem niezależnych zmiennych losowych (i.i.d.), takim że $a_i \leqslant X_i \leqslant b_i$, wtedy:*

$$S_t = \sum_{i=1}^{t} X_i, \quad c_i = b_i - a_i,$$

$$\mathbb{P}(|S_t - \mathbb{E}(S_t)| \geqslant \epsilon) \leqslant 2\exp\left(-\frac{2\epsilon^2}{\sum_{i=1}^{n} c_i^2}\right).$$

Lemat 1.11. *Niech $X_1, X_2, \ldots, X_t$ będzie ciągiem niezależnych zmiennych losowych (i.i.d.), takim że $0 \leqslant X_i \leqslant 1$, wtedy:*

$$\overline{X}_t = \frac{S_t}{t}, \quad \mu = \mathbb{E}(X_i), \quad \mathbb{P}(|\overline{X}_t - \mu| \leqslant \epsilon) = 1 - \delta,$$

$$\epsilon \leqslant \sqrt{\frac{\ln(2/\delta)}{2t}}.$$

Twierdzenie 1.12 (Empiryczna nierówność Bernsteina)**.** *Niech $X_1, X_2, \ldots, X_t$ będzie ciąggiem niezależnych zmiennych losowych (i.i.d.), takim że $a \leqslant X_i \leqslant b$, wtedy:*

$$\mathbb{P}(|\overline{X}_t - \mu| \geqslant \epsilon) \leqslant \delta, \quad \overline{\sigma}_t^2 = \frac{1}{t}\sum_{i=i}^{t}(X_i - \overline{X}_t)^2,$$
$$|\overline{X}_t - \mu| \leqslant \overline{\sigma}_t\sqrt{\frac{2\ln(3/\delta)}{t}} + \frac{3R\ln(3/\delta)}{t}.$$

Lemat 1.13. *Niech $X_1, X_2, \ldots, X_t$ będzie ciąggiem niezależnych zmiennych losowych (i.i.d.), takim że $0 \leqslant X_i \leqslant 1$, wtedy:*

$$\mathbb{P}(|\overline{X}_t - \mu| \leqslant \epsilon) = 1 - \delta,$$
$$\epsilon \leqslant \overline{\sigma}_t\sqrt{\frac{2\ln(3/\delta)}{t}} + \frac{3\ln(3/\delta)}{t}.$$

1.4 Problem porównań wielokrotnych

Załóżmy, że z prawdopodobieństwem $1-\delta$ chcemy ustalić, który z dwóch graczy p_1 i p_2 jest lepszy. W tym celu będziemy przeprowadzać testy statystyczne, dla których prawdopodobieństwo pomyłki k-tego testu wynosi δ_k. Testy te będą kontynuowane aż do momentu, gdy jeden z graczy zwycięży przeważająca liczbę razy. Po przeprowadzeniu n takich testów:

$$\mathbb{P}(\text{Chociaż jeden z } n \text{ testów się pomylił}) \overset{(*)}{\leqslant} \sum_{k=1}^{n}\delta_k \implies$$
$$\mathbb{P}(\text{Żaden test się nie pomylił}) \leqslant 1 - \sum_{k=1}^{n}\delta_k,$$

gdzie nierówność oznaczona $(*)$ wynika z faktu, że $\mathbb{P}(X+Y) \leqslant \mathbb{P}(X) + \mathbb{P}(Y)$.

Aby ostateczne prawdopodobieństwo popełnienia błędu było mniejsze niż δ, konieczne jest wprowadzenie odpowiedniej korekty. Możemy zastosować jedną z dwóch poprawek:

1.4.1 Niech n będzie maksymalną liczbą testów, jaką pozwalamy wykonać, aby wyznaczyć lepszego gracza. Wtedy $\delta_k = \frac{\delta}{n}$.

1.4.2 Niech δ_k spełnia nierówność $\delta \geqslant \sum_{k=1}^{\infty}\delta_k$. Wtedy niezależnie od ilości przeprowadzonych testów, ostateczne prawdopodobieństwo pomyłki będzie nie większe niż δ.

Wykorzystując Lemat 1.13 oraz korektę 1.4.2 otrzymujemy, że dla ciągu $X_1, X_2, \ldots, X_t$ i.i.d. zmiennych losowych, takim że $0 \leqslant X_i \leqslant 1$

$$\epsilon_{t,k} \leqslant \overline{\sigma}_t\sqrt{\frac{2\ln(3/\delta_k)}{t}} + \frac{3\ln(3/\delta_k)}{t}. \tag{1.1}$$

Wtedy $(\overline{X}_t - \epsilon_{t,k}, \overline{X}_t + \epsilon_{t,k})$ interpretujemy jako przedział ufności dla μ o współczynniku ufności $1-\delta_k$.

Lemat 1.14. *Niech $X_1, X_2, \ldots, X_t$ będzie ciąggiem i.i.d. zmiennych losowych, takim że $0 \leqslant X_i \leqslant 1$. Dodatkowo niech liczba rozegranych gier będzie funkcją zależną od k ($f(k) = t$) oraz niech $\delta_k = \frac{\delta}{g(k)}$, gdzie $\delta \geqslant \sum_{k=1}^{\infty}\frac{\delta}{g(k)}$ i $\ln(g(k)) \in o(f(k))$. Wtedy $\lim_{k\to\infty} e_{f(k),k} = 0$.*

Dowód Lematu 1.14 znajduje się na końcu niniejszej pracy w sekcji Dodatek 5.1.

Lemat 1.14 pozwala nam na ograniczenie ilości przeprowadzanych testów do wyznaczenia lepszego gracza oraz określa, w jaki sposób możemy przeprowadzać testy, aby nie stracić zbieżności. Wynik ten jest istotny, ponieważ podejście oparte o próbę wyznaczenia, który z graczy jest lepszy po każdej rozegranej grze prowadzi do nadmiernej ilości wykonywanych testów. Przykładami funkcjami, które możemy użyć w Lematu 1.14 są $f(k) = k^2, g(k) = \frac{6/\pi^2}{k^2}$. Taki dobór funkcji oznacza, że k-ty test odbywa się, gdy liczba przeprowadzonych gier wynosi k^2.

Rozdział 2

Algorytmy wyścigowe

Do algorytmów wykorzystujących korekty 1.4.1 i 1.4.2 należą algorytmy wyścigowe (Racing Algorytms) . Dwoma najpopularniejszymi typami algorytmów racingowych są „Hoeffding race" [6] oraz „Empirical Bernstein race" [7]. Oparte są one odpowiednio o Twierdzenie 1.10 i Twierdzenie 1.13 oraz korektę 1.4.2. Mają one jednak pewną wadę, mogą one wymagać bardzo dużej liczby iteracji, a gdy poziom umiejętności porównywanych graczy jest sobie równy (prawdopodobieństwo wygranej wynosi 50%), wtedy z prawdopodobieństwem równym $1-\delta$ algorytm nigdy się nie zatrzyma.

Aby rozwiązać problemy związane z klasycznymi algorytmami wyścigowymi, wprowadzamy tzw. Limited Racing algorytm. Załóżmy zatem dodatkowy warunek, który mówi, że przerywamy działanie algorytmu, gdy empiryczna wartość oczekiwana z prawdopodobieństwem większym bądź równym $1-\delta$ jest znana z dokładnością co do zadanego ϵ. W naszej pracy przyjmiemy $\epsilon = 0{,}01$ i $\delta = 0{,}05$.

2.1 Limited Empirical Bernstein Race (LEBR) algorytm

Klasyczny algorytm racingowy opiera się na szeregu ϵ_t, spełniającej warunek, że zdarzenie $\mathcal{E} = \{|\overline{X}_t - \mu| \leqslant \epsilon_t, t \in \mathbb{N}^+\}$ występuje z prawdopodobieństwem nie mniejszym niż $1-\delta$. Dodatkowo niech δ_k będzie dodatnim szeregiem spełniającym $\delta \geqslant \sum_{k=1}^{\infty} \delta_k$. Wtedy korzystając z Lematu 1.13

$$\epsilon_t = \overline{\sigma}_t \sqrt{\frac{2\ln(3/\delta_t)}{t}} + \frac{3\ln(3/\delta_t)}{t}.$$

Ponieważ δ_k sumuje się co najwyżej do δ, a $(\overline{X}_t - \epsilon_t, \overline{X}_t + \epsilon_t)$ jest przedziałem ufności dla μ o współczynniku ufności $1-\delta_t$, oznacza to, że zdarzenie $\mathcal{E}$ występuje z prawdopodobieństwem nie mniejszym niż $1-\delta$. Podobny rezultat otrzymujemy stosując ϵ_t oparte o Lemat 1.11, jednak zmienia się wtedy postać ϵ_t. W pracy [1] algorytm opierał się o szereg $\delta_t = \frac{c\delta}{t^2}, c = \frac{6}{\pi^2}$. Pseudokod algorytmu, przedstawiony jako Algorytm 1, opiera się na rozgrywaniu t gier i obliczeniu górnej granicy $\mathrm{UB} = \min_{1\leqslant k\leqslant t}(\overline{X}_k + c_k)$ oraz dolną granice $\mathrm{LB} = \max(0, \max_{1\leqslant k\leqslant t}(\overline{X}_k - c_k))$. Algorytm kończy działanie, gdy różnica między UB a LB jest mniejsza niż 2ϵ. Otrzymane w ten sposób $\overline{X}$ z prawdopodobieństwem nie mniejszym niż $1-\delta$ jest bliskie wartości μ z dokładnością co zadanego ϵ.

Algorytm 1 LEBR

Wprowadź: precyzja ϵ, prawdopodobienstwo δ_k
$LB \leftarrow 0, \quad UB \leftarrow \infty, \quad t \leftarrow 0, \quad n \leftarrow 1$
dopóki $UB - LB > 2\epsilon$ **wykonaj**
$t \leftarrow t + 1$
Realizacja X_t
$\delta_n \leftarrow \frac{6\delta}{\pi^2 n^2}$
$\epsilon_n \leftarrow \overline{\sigma}_n \sqrt{\frac{2\ln(3/\delta_n)}{n}} + \frac{3\ln(3/\delta_n)}{n}$
$LB \leftarrow \max(LB, \overline{X_n} - \epsilon_n)$
$UB \leftarrow \min(UB, \overline{X_n} + \epsilon_n)$
$n \leftarrow n + 1$
koniec dopóki
zwróć $\overline{X_t}$

2.2 Improved LEBR (ILEBR)

Algorytm 1 jest oparty na korekcie 1.4.2, która zakłada możliwie nieskończoną ilość testów.Jednak dodanie ograniczenia odnośnie pożądanej dokładności ϵ pozwala nam określić maksymalną liczbę testów $n_{\max}$, jaką należy wykonać, aby empiryczna wartość oczekiwana $\overline{X}_t$ była równa teoretycznej wartości oczekiwane μ z dokładnością co do ϵ z prawdopodobieństwem nie mniejszym niż $1 - \delta$.

Niech $\delta_k = \frac{\delta}{n_{\max}}$ gdzie $n_{\max}$ oznacza maksymalną ilość testów potrzebną do wyznaczenia lepszego gracza. Rozwiązując numerycznie równanie (2.1) wynikające z Lematu 1.11, możemy ustalić maksymalną potrzebną ilość testów, niezależnie od odchylenia standardowego zmiennej losowej X_i

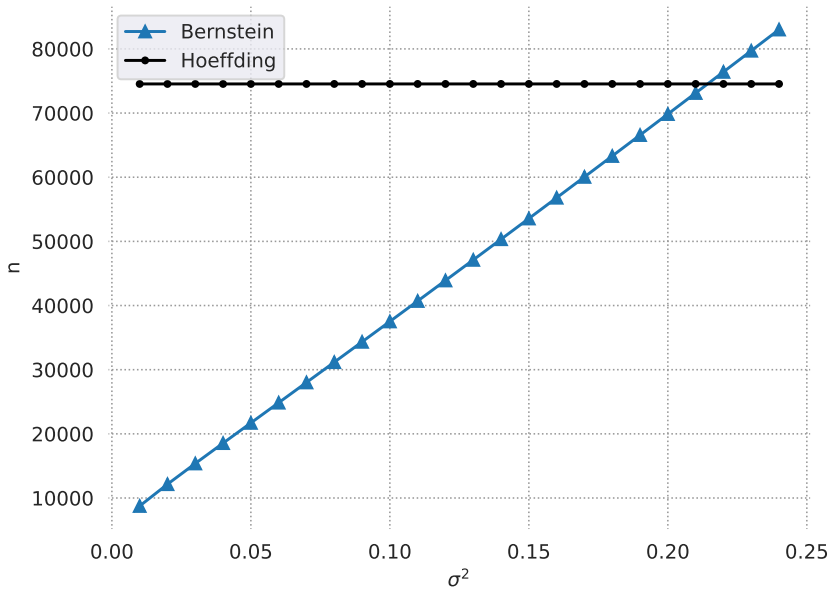
$$\epsilon = \sqrt{\frac{\ln(2n_{\max}/\delta)}{2n_{\max}}}. \tag{2.1}$$

Dla $\epsilon = 0{,}01$ i $\delta = 0{,}05$ rozwiązaniem numerycznym równania (2.1) jest $n_{\max} = 74539{,}85$. Oznacza to, że aby z prawdopodobieństwem nie mniejszym niż $1 - 0{,}05$ oszacować prawdopodobieństwo wygrania gry przez pierwszego gracza z dokładnością co do 0,01, maksymalna ilość gier, jaką należy rozegrać między dwoma graczami wynosi $n_{\max} = 74540$.

Powyższą metodę możemy również zastosować w przypadku nierówności Bernsteina. Na początku jednak musimy oszacować z góry $\overline{\sigma}_t$. Przyjmijmy, że X_i to zmienna losowa o rozkładzie zero-jedynkowym. Wtedy maksymalna możliwa wariancja dla takiej zmiennej losowej wynosi $\sigma^2 = 0{,}25$. Jest to również maksymalna możliwa wartość dla naszej empirycznej wariancji. Podstawiając $\delta_n = \frac{0{,}05}{n_{\max}}, \epsilon = 0{,}01$ do Lematu 1.13 otrzymujemy

$$0{,}01 \leqslant \sqrt{\frac{\ln(\frac{3n_{\max}}{0{,}05})}{2n_{\max}}} + \frac{3\ln(\frac{3n_{\max}}{0{,}05})}{n_{\max}}. \tag{2.2}$$

Dla $\epsilon = 0{,}01$ i $\delta = 0{,}05$ rozwiązaniem numerycznym równania (2.2) jest $n_{\max} = 86329$. Jednak granice oparte o nierówność Bernsteina zależą od wariancji. Sprawdźmy zatem jak wygląda $n_{\max}$ w zależności od σ^2. Z Rysunku 2.1 widzimy, że algorytm oparty o nierówność Bernsteina szybciej kończy działanie w przypadku, gdy jedna z porównywanych strategii jest silnie dominująca. Co więcej, z równania (2.1) wynika, że niezależnie od ilości wykonywanych testów, dla $\delta_n = \frac{0{,}05}{74540}$ maksymalna ilość gier, po której z prawdopodobieństwem $1 - 0{,}05$ wiemy, że $|\overline{X_i} - \mu| \leqslant 0{,}01$ wynosi 74540.



Rysunek 2.1: Wykres maksymalnej potrzebnej ilości testów w zależności od wariancji zmiennych losowych X_i w przypadku, gdy liczba przeprowadzonych testów jest równa liczbie gier ($t = n$) dla $\epsilon = 0{,}01$ i $\delta = 0{,}05$.

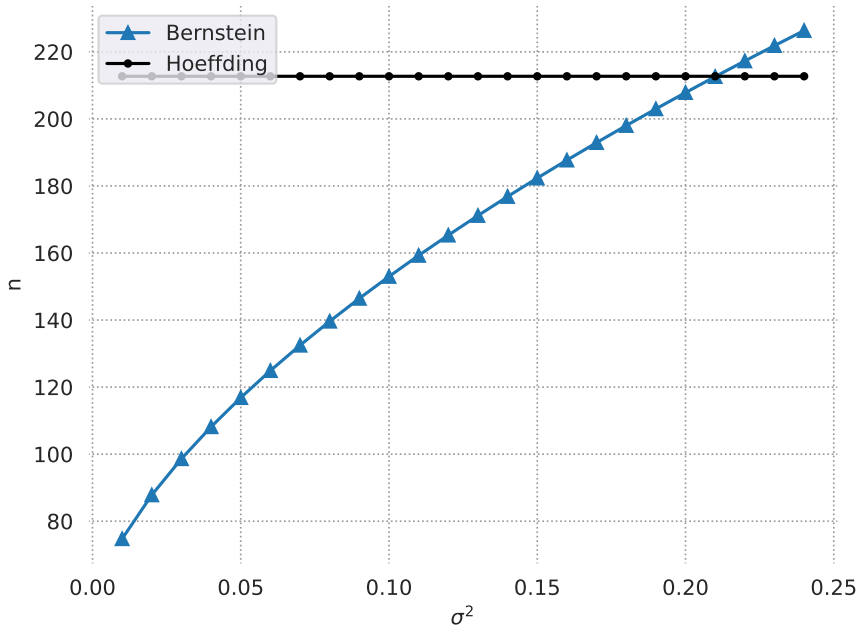
Algorytm 2 jest modyfikacją Algorytmu 1 uwzględniającą ograniczenie, na maksymalną liczbę wykonywanych testów, wynikające z analizy nierówności Bernsteina i Hoeffdinga. W szczególności został dodany warunek zatrzymania algorytmu, gdy liczba wykonanych testów przekroczy $n_{\max}$.

Algorytm 2 ILEBR 1

Wprowadź: precyzja ϵ, prawdopodobienstwo δ
$LB \leftarrow 0, \quad UB \leftarrow \infty, \quad t \leftarrow 0, \quad n \leftarrow 1$
Znajdz $n_{\max}$ takie że $\epsilon = \sqrt{\frac{\ln(2n_{\max}/\delta)}{2n_{\max}}}$
$\delta_n = \delta / n_{\max}$
dopóki $UB - LB > 2\epsilon$ lub $n \leqslant n_{\max}$ **wykonaj**
$t \leftarrow t + 1$
Realizacja X_t
$\epsilon_n \leftarrow \overline{\sigma}_n \sqrt{\frac{2\ln(3/\delta_n)}{n}} + \frac{3\ln(3/\delta_n)}{n}$
$LB \leftarrow \max(LB, \overline{X_n} - \epsilon_n)$
$UB \leftarrow \min(UB, \overline{X_n} + \epsilon_n)$
$n \leftarrow n + 1$
koniec dopóki
zwróć $\overline{X_t}$

2.3 ILEBR 2

Algorytmy 1 i 2 przeprowadzały testy po każdej rozegranej grze, co skutkuje wystąpieniem bardzo dużej ilości testów. Wprowadźmy zatem pewną zmianę do naszych algorytmów. Zamiast przeprowadzać test po każdej rozegranej grze, niech test odbywa się w momencie, gdy



Rysunek 2.2: Wykres maksymalnej potrzebnej ilości testów w zależności od wariancji zmiennych losowych X_i w przypadku, gdy liczba rozegranych gier jest równa liczbie przeprowadzonych testów do kwadratu ($t = n^2$) dla $\epsilon = 0{,}01$ i $\delta = 0{,}05$.

liczba rozegranych gier będzie równa wartości pewnej funkcji zależnej od ilości przeprowadzonych testów. Na podstawie wyników z artykułu [5] wiemy, że funkcją, która pozwoli nam na polepszenie wyników jest $t = n^2$. Aby uzyskać optymalne wyniki, przeprowadźmy tę samą procedurę, jak w przypadku równań (2.1) i (2.2), ale zamiast stosować $t = n$ w Lematach 1.11 i 1.13 zastosujemy $t = n^2$.

W przypadku granicy opartej o nierówność Hoeffdinga otrzymujemy

$$0{,}01 = \sqrt{\frac{\ln(2n_{\max}/\delta)}{2n_{\max}^2}} \implies t_{max} = \lceil n_{\max} \rceil^2 = \lceil 213 \rceil^2 = 45369. \tag{2.3}$$

W przypadku granicy opartej o nierówność Bernsteina otrzymujemy

$$0{,}01 \leqslant \sqrt{\frac{\ln(\frac{3n_{\max}}{0{,}05})}{n_{\max}^2} + \frac{3\ln(\frac{3n_{\max}}{0{,}05})}{n_{\max}^2}} \implies t_{max} = \lceil n_{\max} \rceil^2 = \lceil 230{,}751^2 \rceil = 53247. \tag{2.4}$$

Porównując uzyskane wartości (2.3), (2.4) z wartościami (2.1) i (2.2), widzimy, że udało się nam zmniejszyć maksymalną ilość gier, które należy wykonać z 74540 do 45369. Wprowadźmy więc nowe poprawki do Algorytmu 3.

2.4 ILEBR*

Do tej pory wszystkie stosowane algorytmy wyznaczały nam prawdopodobieństwo wygranej pierwszego gracza. Jednak nam nie zależy na tym, aby dokładnie oszacować prawdopodobieństwo wygranej. Głównym celem algorytmów jest wyznaczenie lepszej strategii.

Algorytm 3 ILEBR 2

Wprowadź: precyzja ϵ, prawdopodobienstwo δ
$LB \leftarrow 0, \quad UB \leftarrow \infty, \quad t \leftarrow 0, \quad n \leftarrow 1$
Znajdz $n_{\max}$ takie że $\epsilon = \sqrt{\frac{\ln(2n_{\max}/\delta)}{2n_{\max}^2}}$
$\delta_n = \delta / n_{\max}$
dopóki $UB - LB > 2\epsilon$ lub $n \leqslant n_{\max}$ **wykonaj**
  dopóki $t \leqslant n^2$ **wykonaj**
    $t \leftarrow t + 1$
    Realizacja X_t
  koniec dopóki
  $\epsilon_n \leftarrow \overline{\sigma}_n \sqrt{\frac{2\ln(3/\delta_n)}{n^2}} + \frac{3\ln(3/\delta_n)}{n^2}$
  $LB \leftarrow \max(LB, \overline{X_n} - \epsilon_n)$
  $UB \leftarrow \min(UB, \overline{X_n} + \epsilon_n)$
  $n \leftarrow n + 1$
koniec dopóki
zwróć $\overline{X_t}$

Pozwala nam to na modyfikacje Algorytmu 3 o nowe warunki zatrzymania. Warunkiem tym jest zatrzymanie działania algorytmu w momencie, gdy górna bądź dolna granica przekroczy wartość 0,5.

Algorithm 4: ILEBR* 1

Wprowadź: precyzja ϵ, prawdopodobienstwo δ
$LB \leftarrow 0, \quad UB \leftarrow \infty, \quad t \leftarrow 1, \quad n \leftarrow 0$
Znajdz $n_{\max}$ takie że $\epsilon = \sqrt{\frac{\ln(2n_{\max}/\delta)}{2n_{\max}^2}}$
$\delta_n = \delta / n_{\max}$
dopóki ($UB - LB > 2\epsilon$ lub $n \leqslant n_{\max}$) i ($UB > 0{,}5$ lub $LB < 0{,}5$) **wykonaj**
  dopóki $t \leqslant n^2$ **wykonaj**
    $t \leftarrow t + 1$
    Realizacja X_t
  koniec dopóki
  $\epsilon_n \leftarrow \overline{\sigma}_n \sqrt{\frac{2\ln(3/\delta_n)}{n^2}} + \frac{3\ln(3/\delta_n)}{n^2}$
  $LB \leftarrow \max(LB, \overline{X_n} - \epsilon_n)$
  $UB \leftarrow \min(UB, \overline{X_n} + \epsilon_n)$
  $n \leftarrow n + 1$
koniec dopóki
jeżeli $LB > 0{,}5$ **to**
  zwróć p_1 lepszy
jeśli jednak $UB < 0{,}5$ **to**
  zwróć p_2 lepszy
jeśli jednak $\overline{X_n} > 0{,}5$ **to**
  zwróć p_1 lepszy
w przeciwnym razie
  zwróć p_2 lepszy
koniec jeżeli

2.5 ILEBR* 2

Do tej pory skupialiśmy się na ograniczeniu maksymalnej ilości rozgrywanych gier. Jednak oczywiste jest, że nie ma sensu testować, który z graczy jest lepszy, gdy ilość rozegranych

gier jest zbyt mała. Jednak jak wyznaczyć nasze minimalne t, po którym przeprowadzamy pierwszy test? Algorytm oparty na nierówności Bernsteina najszybciej wyznacza wynik, gdy wariancja naszej zmiennej losowej wynosi 0 (jeden z graczy zawsze wygrywa). Dla Algorytmu 4 interesuje nas moment, od którego będziemy w stanie rozróżnić, kiedy dolna lub górna granica przekroczy 0,5. Zatem oszacujmy n_{min} korzystając z Lematu 1.13 dla $\overline{\sigma}_t = 0$ i $t = n^2$.

$$0{,}5 \leqslant \frac{3\ln(\frac{3n_{\max}}{0{,}05})}{n_{min^2}} \implies t_{min} = \lceil n_{min} \rceil^2 = \lceil 7{,}53219 \rceil^2 = 64.$$

Oznacza to, że przy $\epsilon = 0{,}01$ i $\delta = 0{,}05$ w przypadku, gdy jeden gracz zawsze wygrywa, będziemy w stanie to stwierdzić nie wcześniej niż po rozegraniu 64 gier. Na tej podstawie wyznaczmy nowe $n_{\max}$ uwzględniając pomijanie pierwszych niepotrzebnych testów.

$$0{,}01 = \sqrt{\frac{\ln(2n_{\max}/0{,}05)}{2(n_{\max}+7)^2}} \implies t_{max} = \lceil n_{\max} + 6 \rceil^2 = \lceil 205{,}289 + 7 \rceil^2 = 45369. \tag{2.5}$$

Chociaż wynik równania (2.5) nie zmniejszył maksymalnej liczby testów jakie należy wykonać, to pozwolił nam on na zmniejszenie ϵ_n. Oznacza to, że stosując odroczenie pierwszych testów, możemy dokładniej oszacować naszą górną i dolną granicę w stosowanych algorytmach.

Algorithm 5: ILEBR* 2

Wprowadź: precyzja ϵ, prawdopodobienstwo δ
$LB \leftarrow 0, \quad UB \leftarrow \infty, \quad n \leftarrow 0$
Znajdz $n_{\max}$ takie że $\epsilon = \sqrt{\frac{\ln(2n_{\max}/\delta)}{2n_{\max}^2}}$
Znajdz n_{min} takie że $0{,}5 = \sqrt{\frac{\ln(2n_{\max}/\delta)}{2(n_{min}+1)^2}}$
Realizacja pierwszych $X_1, X_2, X_{n_{min}}$ gier
$t \leftarrow n_{min}^2$
$\delta_n = \delta / n_{\max}$
dopóki ($UB - LB > 2\epsilon$ lub $n \leqslant n_{\max}$) i ($UB > 0{,}5$ lub $LB < 0{,}5$) **wykonaj**
  dopóki $t \leqslant (n + n_{min})^2$ **wykonaj**
    $t \leftarrow t + 1$
    Realizacja X_t
  koniec dopóki
  $\epsilon_n \leftarrow \overline{\sigma}_n \sqrt{\frac{2\ln(3/\delta_n)}{(n+n_{min})^2}} + \frac{3\ln(3/\delta_n)}{(n+n_{min})^2}$
  $LB \leftarrow \max(LB, \overline{X_n} - \epsilon_n)$
  $UB \leftarrow \min(UB, \overline{X_n} + \epsilon_n)$
  $n \leftarrow n + 1$
koniec dopóki
jeżeli $LB > 0{,}5$ **to**
  zwróć p_1 lepszy
jeśli jednak $UB < 0{,}5$ **to**
  zwróć p_2 lepszy
jeśli jednak $\overline{X_n} > 0{,}5$ **to**
  zwróć p_1 lepszy
w przeciwnym razie
  zwróć p_2 lepszy
koniec jeżeli

2.6 Prawdopodobieństwo popełnienia błędu

Algorytmy typu LEBR charakteryzują się odmiennym prawdopodobieństwem popełnienia błędu niż klasyczne algorytmy racingowe. Wynika to z wprowadzenia parametru ϵ.

W tej sekcji zajmiemy się określeniem tego prawdopodobieństwa w zależności od parametrów początkowych δ i ϵ w badanym problemie decyzyjnym. W tym celu zaproponujemy dwie hipotezy dotyczące graczy:

- H_a: Gracz pierwszy jest lepszy od gracza drugiego.
- H_b: Gracz pierwszy jest gorszy od gracza drugiego.

To oznacza, że prawdopodobieństwem popełnienia błędu w naszym przypadku jest prawdopodobieństwo uznania za lepszego gracza osoby o mniejszym prawdopodobieństwie zwycięstwa.

Oznaczmy prawdopodobieństwo pomyłki w naszych algorytmach typu LEBR jako α. Bez straty ogólności załóżmy, że teoretyczne prawdopodobieństwo wygrania gry przez pierwszego gracza jest większe od 0,5 ($\mu > 0{,}5$). Wtedy prawdopodobieństwo α możemy wyrazić za pomocą następującego wzoru

$$\alpha = 1 - \mathbb{P}_\mu(\overline{X}_{f(n_{\max})} > 0{,}5) \prod_{k=1}^{n_{\max}} \mathbb{P}_\mu(\overline{X}_{f(k)} + \epsilon_k > 0{,}5), \tag{2.6}$$

gdzie:

- α: Prawdopodobieństwo popełnienia błędu.
- $f(k)$: Funkcja rozmiaru próby, zależna od liczby przeprowadzonych testów.
- $\epsilon_k = \overline{\sigma}_k \sqrt{\frac{2\ln(3/\delta_k)}{f(k)}} + \frac{3\ln(3/\delta_k)}{f(k)}$: Zależny od testowanego algorytmu.
- $n_{\max}$: Zależny od przyjętych parametrów ϵ i δ.

Dowód wyrażenia (2.6) znajduje się na końcu niniejszej pracy w Dodatku 5.2.

Wyrażenie (2.6) pozwala nam określić teoretyczne prawdopodobieństwo wystąpienia błędu dla algorytmów typu LEBR. Niestety, wzór ten nie umożliwia nam na łatwe wyznaczenie takiej wartości parametru δ, aby prawdopodobieństwo błędu było mniejsze niż żądane α. Aby uzyskać metodę wyznaczania takiej δ dla zadanego parametru początkowego ϵ, ograniczmy z góry prawdopodobieństwo błędu α wynikające z równania (2.6).

$$\alpha \leqslant 1 - \prod_{k=1}^{n_{\max}} \mathbb{P}_\mu(\overline{X}_{f(k)} + \epsilon_k > 0{,}5) + \mathbb{P}_\mu(\overline{X}_{f(n_{\max})} \leqslant 0{,}5). \tag{2.7}$$

Wyrażenie (2.7) możemy zapisać przy pomocy dwóch składników. Pierwszym z nich jest

$$\alpha_1 = \mathbb{P}_\mu(\overline{X}_{f(n_{\max})} \leqslant 0{,}5). \tag{2.8}$$

Oraz drugi składnik

$$\alpha_2 = 1 - \prod_{k=1}^{n_{\max}} \mathbb{P}_\mu(\overline{X}_{f(k)} + \epsilon_k > 0{,}5). \tag{2.9}$$

Zakładając, że X_i jest zmienną losową z rozkładu zero-jedynkowego, możemy stwierdzić, że $Y_n = \sum_{i=1}^{n} X_i$ jest zmienną losową o rozkładzie dwumianowym. To pozwala nam zapisać równania (2.8) i (2.9) w następujący sposób:

$$\alpha_1 = \mathbb{P}_\mu \left(Y_{f(n_{\max})} \leqslant \frac{f(n_{\max})}{2} \right), \tag{2.10}$$

$$\alpha_2 = 1 - \prod_{k=1}^{n_{\max}} \mathbb{P}_\mu \left(Y_{f(n_{\max})} > \frac{f(k)}{2} - \overline{\sigma}_{f(k)} \sqrt{2 \ln(3/\delta_k) f(k)} - 3 \ln(3/\delta_n) \right). \tag{2.11}$$

Analizując Rysunek 2.3 oraz wyrażenie (2.11) możemy zauważyć, że maksymalne α_2 jest osiągane dla $\mu = 0{,}5$, zatem

$$\alpha_2 \leqslant \prod_{k=1}^{n_{\max}} \mathbb{P}_{0,5} \left(Y_{f(n_{\max})} > \frac{f(k)}{2} - \sqrt{\frac{\ln(3/\delta_k) f(k)}{2}} - 3 \ln(3/\delta_n) \right). \tag{2.12}$$

Nierówność (2.12) możemy jeszcze bardziej ograniczyć ze względu na fakt, że $\delta \in (0, 1]$. Analizując Rysunek 2.3 i nierówność (2.12), możemy zauważyć, że wartość prawdopodobieństwa α_2 rośnie wraz ze wzrostem parametru δ. Oznacza to, że możemy zwiększyć nasze ograniczenie, zastępując $\delta_n = \delta / n_{\max}$ samym $1/n_{\max}$.

$$\alpha_2 \leqslant \prod_{k=1}^{n_{\max}} P_{0,5} \left(Y_{f(n_{\max})} > \frac{f(k)}{2} - \sqrt{\frac{\ln(3n_{\max}) f(k)}{2}} - 3 \ln(3n_{\max}) \right). \tag{2.13}$$

Wynik uzyskany z nierówności (2.13) jest szczególnie istotny, ponieważ ograniczenie to zależny jawnie tylko od parametrów μ i $n_{\max}$.

W celach uproszenia zapisy wprowadźmy nową funkcję $F'_\mu(n_{\max})$

$$F'_{\alpha_1}(n_{\max}) = P_\mu \left(Y_{f(n_{\max})} \leqslant \frac{f(n_{\max})}{2} \right), \tag{2.14}$$

$$F'_{\alpha_2}(n_{\max}) = \prod_{k=1}^{n_{\max}} P_{0,5} \left(Y_{f(k)} > \frac{f(k)}{2} - \sqrt{\frac{\ln(3n_{\max})}{2}} f(k) - 3 \ln(3n_{\max}) \right), \tag{2.15}$$

$$F'_\mu(n_{\max}) = 1 - F'_{\alpha_1}(n_{\max}) F'_{\alpha_2}(n_{\max}). \tag{2.16}$$

Łącząc wyniki uzyskane z (2.7) i (2.16) otrzymujemy, że dla $\mu > 0{,}5$

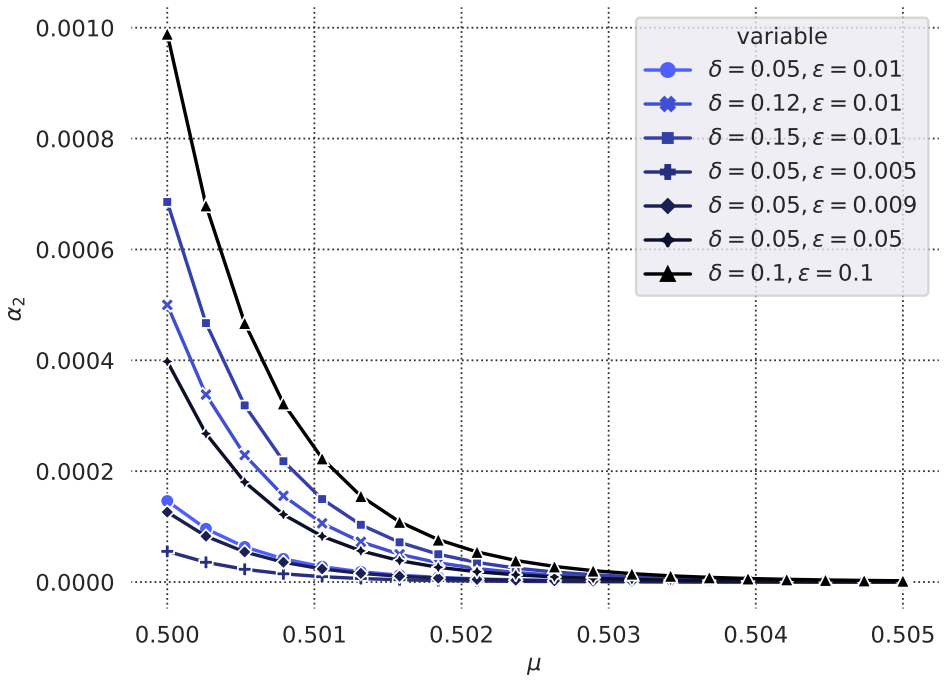
$$F'_\mu(n_{\max}) \geqslant \alpha. \tag{2.17}$$

Ze względu na symetrie rozważanego problemu, nierówność (2.17) możemy zapisać dla dowolnego μ w następujący sposób

$$F_\mu(n_{\max}) = \begin{cases} F'_{1-\mu}(n_{\max}), & \text{dla } \mu \leqslant 0{,}5, \\ F'_\mu(n_{\max}), & \text{dla } \mu > 0{,}5. \end{cases} \tag{2.18}$$

Funkcja (2.18) pozwala nam na określenie prawdopodobieństwa popełnienia błędu dla algorytmów typu LEBR. Ponadto, przy zadanej wartości początkowej parametru ϵ możemy dobrać wartość δ tak, aby nierówność $F_\mu(n_{\max}) \geqslant \alpha$ była spełniona. Takie δ możemy wyznaczyć poprzez znalezienie rozwiązania układu równań

$$\begin{cases} n_{\max} = \min \left\{ n \in \mathbb{N}^+ : F_\mu(n) < \alpha \right\}, \\ \delta = \dfrac{2 n_{\max}}{\exp(2\epsilon^2 f(n_{\max}))}. \end{cases} \tag{2.19}$$



Rysunek 2.3: Wykres wartości prawdopodobieństwa α_2 w zależności od μ, δ i ϵ dla algorytmu ILEBR*.

Dowód wyrażenia (2.19) znajduje się na końcu niniejszej pracy w Dodatku 5.3.

W ramach sprawdzenia wyznaczmy prawdopodobieństwo wyznaczenia złego gracza dla Algorytmu ILEBR* i parametrów początkowych $\mu = 0{,}497, \delta = 0{,}05, \epsilon = 0{,}01$. Wtedy:

$$n_{\max} = 213, \quad \epsilon_n = \overline{\sigma}_n \sqrt{\frac{2\ln(3/\delta_n)}{n^2}} + \frac{3\ln(3/\delta_n)}{n^2}, \quad f(n) = n^2, \quad \delta_n = \frac{\delta}{n_{\max}}. \tag{2.20}$$

Podstawiając (2.20) do (2.18) otrzymujemy, że

$$F_{0{,}497}(213) \approx 0{,}10086 \geqslant \alpha \tag{2.21}$$

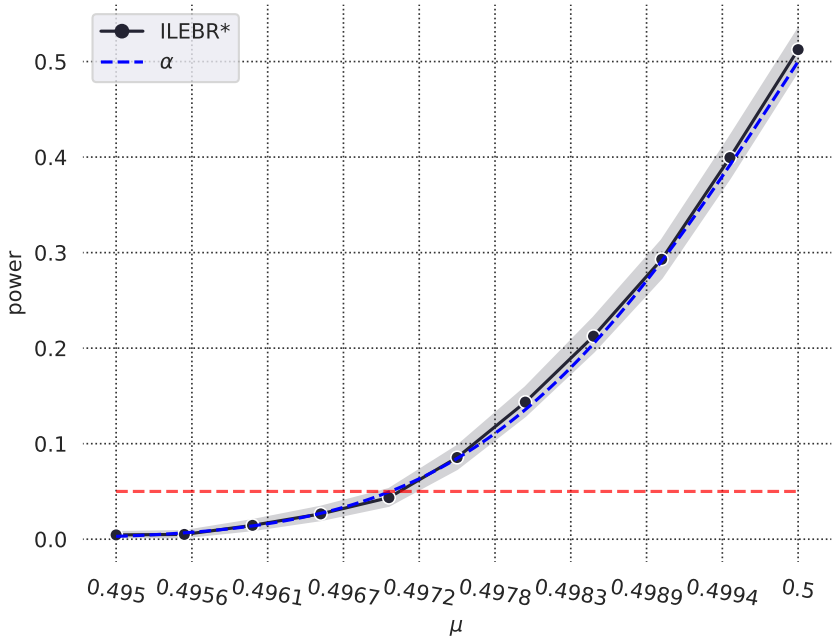
Wynik (2.21) zgadza się z wynikami symulacji dla algorytmu ILEBR* przedstawionymi na Rysunku 2.5.

Co więcej, możemy również wyznaczyć wartość parametru δ tak, aby $F_{0{,}497}(n_{\max}) \leqslant 0{,}05$. Aby to osiągnąć, znajdziemy $n_{\max}$ i δ spełniające zależność (2.19) przy ustalonym $\epsilon = 0{,}01$ i $\alpha = 0{,}05$.

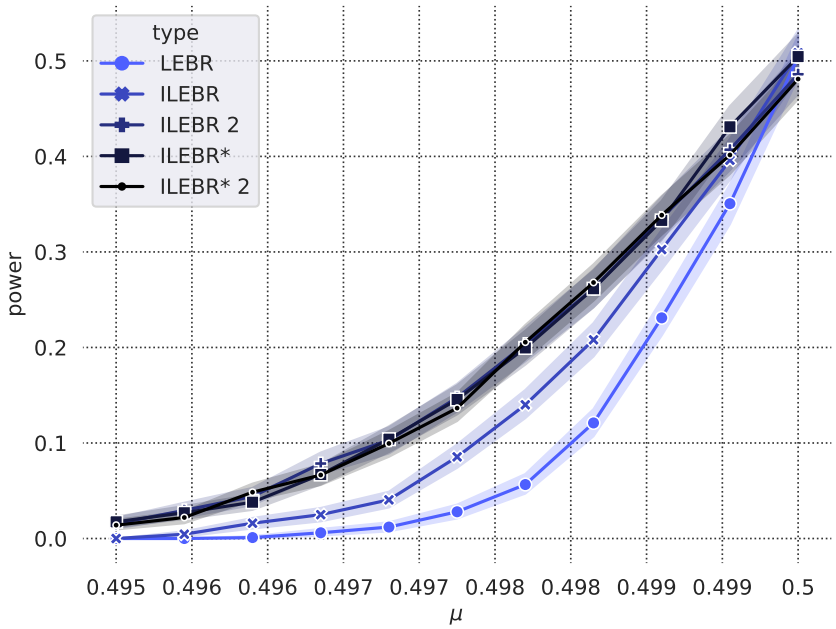
Rozwiązaniem tego problemu przy zadanych parametrach początkowych jest $n_{\max} = 275, \delta = 0{,}00014848$. Rysunek 2.4 przedstawia zgodność symulacji z naszymi teoretycznymi rozważaniami.

Patrząc na Rysunek 2.5 widzimy, że wszystkie testy charakteryzują się niskim prawdopodobieństwem popełnienia błędu i sprawdzają się dobrze, dopóki prawdopodobieństwo wygranej jednego z graczy nie przekroczy 0,495 przy zadanych wartościach parametrach początkowych. Obszary ściemnione oznaczają przedziały ufności o współczynniku ufności 95%.

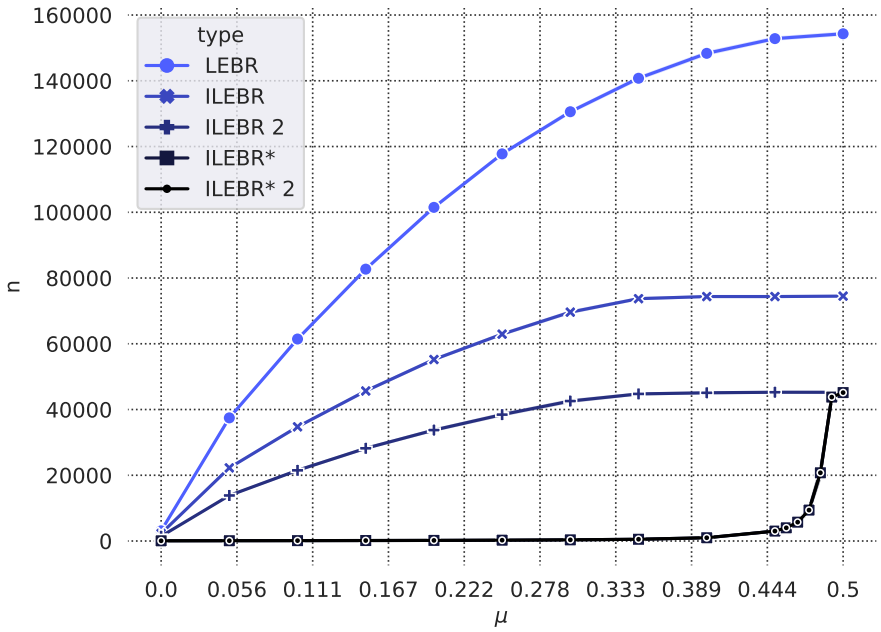
Porównując ze sobą wyniki przedstawione na Rysunkach 2.5 i 2.6 możemy zauważyć, że algorytm LEBR charakteryzuje się najniższym prawdopodobieństwem popełnienia błędu wśród badanych metod. Jednak by osiągnąć taki rezultat, algorytm ten wymaga użycia większej liczby gier, niż pozostałe testowane metody.



Rysunek 2.4: Wykres wartości prawdopodobieństwa α w zależności od μ dla $\delta = 0{,}00014848$, $\epsilon = 0{,}01$ i algorytmu ILEBR*.



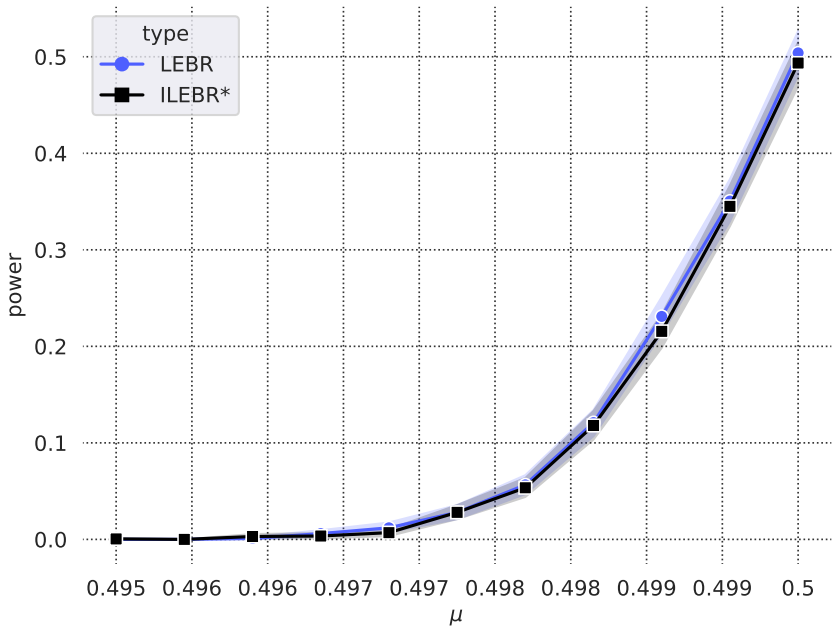
Rysunek 2.5: Wykres prawdopodobieństwa pomyłki w zależności od μ dla $\epsilon = 0{,}01$ i $\delta = 0{,}05$.



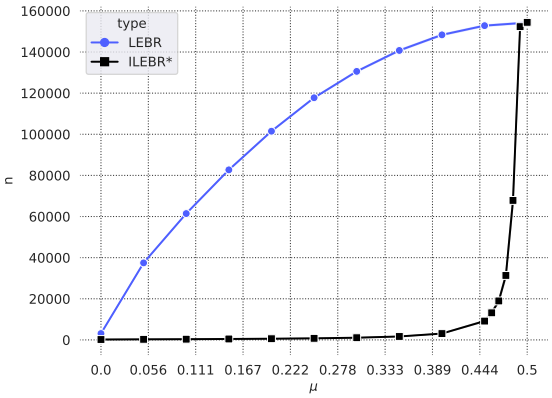
Rysunek 2.6: Wykres średniej liczby gier potrzebnych do rozegrania w zależności od wartości oczekiwanej zmiennych losowych X_i dla $\epsilon = 0{,}01$ i $\delta = 0{,}05$. Na wykresie widoczne są 4 krzywe, ponieważ wyniki dla algorytmów ILEBR* i ILEBR* 2 pokrywają się ze sobą.

Test statystyczny oparty o algorytm ILEBR ma gorsze prawdopodobieństwo, ale pozwala nam na prawie dwukrotnie zmniejszenie liczby wymaganych gier. Dodatkowo widzimy, że algorytmy ILEBR 2, ILEBR* i ILEBR* 2 cechują się takim samym prawdopodobieństwem pomyłki, jednak wersje algorytmów oznaczone (*) pozwalają nam na ograniczenie liczby gier potrzebnej do rozegrania, aby test mógł wyznaczyć lepszego gracza. W poniższej pracy będziemy stosować algorytm ILEBR* 2, ponieważ pozwoli on nam na znaczne zmniejszenie liczby porównań, jakie będziemy przeprowadzać w celu wyznaczenia lepszego gracza.

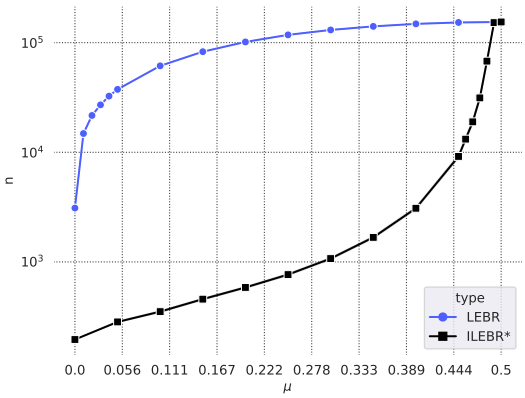
Jeśli jednak stanowi dla nas problem, że algorytmy oznaczone (*) charakteryzują się większym prawdopodobieństwem popełnienia błędu w porównaniu do algorytmu LEBR przy takich samych parametrach początkowych, to możemy łatwo osiągnąć takie samo prawdopodobieństwo, wystarczy, aby $n_{\max}$ w obu tych algorytmach były sobie równe. Analizując Rysunek 2.7 widzimy, że przy zachowaniu tego samego prawdopodobieństwa pomyłki, algorytm ILEBR* jest znacznie szybszy niż jego oryginalny odpowiednik. W naszym przypadku algorytm ILEBR* z parametrami początkowymi $\epsilon = 0{,}01, \delta = 3{,}02101484 * 10^{-11}$ zachowuje taką samą moc jak algorytm LEBR z parametrami początkowymi $\epsilon = 0{,}01, \delta = 0{,}05$.



(a) Wykres prawdopodobieństwa pomyłki w zależności od μ dla $\epsilon = 0{,}01$.



(b) Wykres średniej liczby gier potrzebnej do rozegrania w zależności od μ dla $\epsilon = 0{,}01$.



(c) Wykres średniej liczby gier potrzebnej do rozegrania w zależności od μ dla $\epsilon = 0{,}01$ w skali logarytmicznej.

Rysunek 2.7: Porwanie algorytmów LEBR oraz ILEBR* w przypadku, gdy średnie $n_{\max}$ są sobie równe.

Chapter 3

Algorytmy wyznaczania optymalnej strategii

Wszystkie algorytmy przedstawione w tym rozdziale są algorytmami genetycznymi [4]. Algorytm genetyczny to sposób tworzenia nowych rozwiązań dla danego problemu poprzez iteracyjne stosowanie procesu ewolucyjnego. W tym procesie tworzone są nowe rozwiązania, ocenia się ich jakość i wybierane najlepsze z nich, aby stworzyć kolejną generację rozwiązań. Ten proces powtarza się, aż zostanie znalezione rozwiązanie spełniające określone kryteria. Algorytm generacyjny może być używany do rozwiązywania różnych rodzajów problemów, takich jak optymalizacja, uczenie maszynowe, tworzenie sztucznej inteligencji i wiele innych. Algorytmy ewolucyjne są często interpretowane jako odwzorowanie procesu ewolucyjnego w naturze, ze względu na swoje upodobnienie do procesu selekcji naturalnej, w którym najlepsze jednostki są wybierane do reprodukcji i tworzenia nowych pokoleń.

Ogólnie rzecz biorąc, algorytm generacyjny składa się z kilku kluczowych kroków:

- Inicjalizacja: Tworzenie początkowego zbioru rozwiązań dla danego problemu.
- Ocena: Ocena jakości każdego z rozwiązań za pomocą odpowiedniej funkcji celu lub innych miar.
- Selekcja: Wybieranie najlepszych rozwiązań do następnej generacji.
- Krzyżowanie: Łączenie najlepszych rozwiązań z poprzedniej generacji, aby stworzyć nowe rozwiązania dla następnej generacji.
- Mutacja: Losowa zmiana jednego lub więcej elementów w nowych rozwiązaniach, aby zapewnić różnorodność w następnej generacji.
- Powtarzanie: Powtarzanie kroków 2-5, aż zostanie znalezione rozwiązanie spełniające określone kryteria jakości lub osiągnięty zostanie maksymalny poziom iteracji.

Proces znajdowania potencjalnych rozwiązań polega na przeszukiwaniu przestrzeni wszystkich możliwych rozwiązań i wybieraniu tych, które dają najlepsze wyniki. W rzeczywistości jednak często nie mamy fizycznej możliwości sprawdzenia wszystkich możliwych rozwiązań lub ich sprawdzenie jest zbyt czasochłonne bądź kosztowne. Dlatego w algorytmach ewolucyjnych często wykorzystuje się techniki probabilistyczne, które pomagają wybierać, tworzyć i wyszukiwać kolejne rozwiązania.

W niniejszej pracy zaprezentujemy 4 algorytmy, których celem jest znalezienie optymalnej strategii. Trzy pierwsze algorytmy zostaną opisane na podstawie pracy [1]. Czwartym

algorytmem jest proponowana przeze mnie metoda, która jest bardzo podobna do Algorytmu 8. Opis tego algorytmu zostanie przedstawiony w dalszej części pracy.

3.1 Algorytm iteracyjny

Pierwszym algorytmem, który zostanie przedstawiony, jest algorytm iteracyjny. Jest to metoda bardzo intuicyjna, opierająca się na stopniowym zwiększaniu skuteczności strategii graczy poprzez porównywanie ich wyników. Gdy znajdziemy strategię, która daje lepsze wyniki niż ta poprzednia, staje się ona nowym punktem odniesienia (baseline). Na jej podstawie algorytm będzie kontynuować proces optymalizacji wyników graczy. Nowa strategia jest akceptowana, jeśli wygrywa ona z prawdopodobieństwem większym niż 50% w porównaniu do poprzedniej strategii. Algorytm 6 jest przykładem algorytmu ewolucyjnego typu (1+1) [3].

Algorithm 6: Iterative algorithm

Wprowadź: precyzja ϵ, prawdopodobienstwo δ, losowy przeciwnik x
  $\sigma \leftarrow 1$
  dopóki (kryterium zakończenia nie jest spełnione) **wykonaj**
    dla każdego $i = 1$ do długości x **wykonaj**
      $x'_i \leftarrow x_i + \sigma\mathcal{N}(0,1)$
    koniec dla każdego
    dopóki ILEBR* 2 trawa **wykonaj**
      rozegraj grę pomiędzy x' i x
    koniec dopóki
    jeżeli x' lepszy niż x **to**
      $x \leftarrow x'$
      $\sigma \leftarrow 1{,}25\sigma$
    w przeciwnym razie
      $\sigma \leftarrow 0{,}84\sigma$
    koniec jeżeli
  koniec dopóki
  zwróć przybliżoną strategię optymalna x

3.2 Real Coevolution algorytm

Kolejnym algorytmem ewolucyjnym, którego będziemy używać, jest algorytm koewolucyjny. W przypadku tej metody nowy punkt odniesienia jest wybierany w momencie, gdy nowo znaleziona strategia okazuje się lepsza niż wszystkie dotychczas wybrane. Pseudokod algorytmu koewolucyjnego został przedstawiony jako Algorytm 7.

3.3 Approx Coevolution 1 algorytm

W przypadku dwóch poprzednich algorytmów nowe rozwiązanie jest tworzone na podstawie poprzednio znalezionego rozwiązania. Możemy jednak zastosować tzw. "podejście Paryskie" (Parisian approach)[2]. W tym podejściu, zamiast porównywać nowo uzyskaną strategię z każdą poprzednio przyjętą, porównujemy ją tylko z jedną losowo wybraną strategią z populacji. Pseudokod algorytmu koewolucyjnego został przedstawiony jako Algorytm 8.

Algorithm 7: Real Coevolution

Wprowadź: precyzja ϵ, prawdopodobienstwo δ, random opponent x
$\sigma \leftarrow 1$
$P \leftarrow \{x\}$
dopóki (kryterium zakończenia nie jest spełnione) **wykonaj**
  dla każdego $i = 1$ do długości x **wykonaj**
    $x'_i \leftarrow x_i + \sigma\mathcal{N}(0,1)$
  koniec dla każdego
  dla każdego $i = 1$ do długości P **wykonaj**
    dopóki ILEBR* 2 trawa **wykonaj**
      rozegraj grę pomiędzy x' i P_i
    koniec dopóki
  koniec dla każdego
  jeżeli x' lepszy niż wszystkie punkty z P **to**
    $x \leftarrow x'$
    $P \leftarrow \{P,x'\}$
    $\sigma \leftarrow 1{,}25\sigma$
  w przeciwnym razie
    $\sigma \leftarrow 0{,}84\sigma$
  koniec jeżeli
koniec dopóki
zwróć

Algorithm 8: Approximate Coevolution

Wprowadź: precyzja ϵ, prawdopodobienstwo δ, losowy przeciwnik x
$\sigma \leftarrow 1$
$P \leftarrow \{x\}$
dopóki (kryterium zakończenia nie jest spełnione) **wykonaj**
  dla każdego $i = 1$ do długości x **wykonaj**
    $x'_i \leftarrow x_i + \sigma\mathcal{N}(0,1)$
  koniec dla każdego
  k ← liczba naturalna z przedziału od 1 do długości P
  dopóki ILEBR* 2 trawa **wykonaj**
    rozegraj grę pomiędzy x' i P_k
  koniec dopóki
  jeżeli x' lepszy **to**
    $x \leftarrow x'$
    $P \leftarrow \{P,x'\}$
    $\sigma \leftarrow 1{,}25\sigma$
  w przeciwnym razie
    $\sigma \leftarrow 0{,}84\sigma$
  koniec jeżeli
koniec dopóki
zwróć

3.4 Approx Coevolution 2 algorytm

Ostatnim algorytmem, którym się zajmiemy, jest algorytm koewolucyjny. Tym razem, zamiast porównywać naszą strategię z jedną losowo wybraną strategią z populacji, tak jak to robiliśmy w przypadku Algorytmu 8, nasza strategia będzie testowana przeciwko losowo wybranemu przeciwnikowi. Pseudokod algorytmu koewolucyjnego został przedstawiony jako Algorytm 9.

Algorithm 9: Approximate Coevolution 2

Wprowadź: precyzja ϵ, prawdopodobienstwo δ, losowy przeciwnik x
$\sigma \leftarrow 1$
$P \leftarrow \{x\}$
dopóki (kryterium zakończenia nie jest spełnione) **wykonaj**
  dla każdego $i = 1$ do długości x **wykonaj**
    $x'_i \leftarrow x_i + \sigma\mathcal{N}(0,1)$
  koniec dla każdego
  dopóki ILEBR* 2 trawa **wykonaj**
    rozegraj grę pomiędzy x' i losową strategia z P
  koniec dopóki
  jeżeli x' lepszy **to**
    $x \leftarrow x'$
    $P \leftarrow \{P,x'\}$
    $\sigma \leftarrow 1{,}25\sigma$
  w przeciwnym razie
    $\sigma \leftarrow 0{,}84\sigma$
  koniec jeżeli
koniec dopóki
zwróć przybliżoną strategię optymalna x

Chapter 4

Gry

Aby sprawdzić poprawność działania powyższych algorytmów przetestujemy je na podstawie dwóch gier. Pierwszą z nich będzie popularna gra karciana zwana Wojną. Drugą grą, w której będziemy szukać optymalnej strategii, będzie gra o nazwie Rrrats. Przedstawmy najpierw zasady obowiązujące w obu tych grach.

4.1 Gra w wojnę

Wojna to prosta gra karciana, w której uczestnicy grają przeciwko sobie i używają talii standardowych kart do gry. Celem gry jest zdobycie wszystkich kart od przeciwnika.

Zasady gry są następujące:

- Gracze rozdają po 26 kart, tak aby każdy miał swoje ukryte "magazyny".
- Następnie jedna karta jest odkrywana z każdego magazynu i porównywana ze sobą. Gracz, który ma kartę o wyższej wartości, zabiera obie karty i dokłada je na koniec swojego magazynu. Jeśli karty są takie same, gracze rozgrywają "wojnę".
- Przy wojnie, obaj gracze wykładają z magazynów najpierw jedną kartę rewersem do góry, a następnie odkrywają kolejną kartę rewersem do dołu. Ten gracz, który ma kartę o wyższej wartości, zabiera wszystkie karty i dokłada je do swojego magazynu. Jeśli karty są takie same, proces powtarza się, aż do momentu, gdy jeden z graczy wygra.
- Gra kończy się w momencie, gdy któryś z graczy wygra wszystkie karty.

Same zasady gry nie definiują jednak tego, w jaki sposób karty na koniec naszego "magazynu" mogą zostać umieszczane.

Wprowadźmy zatem 3 parametry, nazwijmy je odpowiednio A, B, C. Będziemy używać ich do wyznaczania nieujemnych parametrów $\alpha = \exp(A),\ \beta = \exp(B), \gamma = \exp(C)$. Po wygranej turze umieśćmy k zdobytych kart na końcu naszego "magazynu" w następujący sposób:

- karty w kolejności malejącej z prawdopodobieństwem równym $\alpha/(\alpha + \beta + \gamma)$
- karty w kolejności rosnącej z prawdopodobieństwem równym $\beta/(\alpha + \beta + \gamma)$
- karty w kolejności losowej z prawdopodobieństwem równym $\gamma/(\alpha + \beta + \gamma)$

4.2 Rrrats!

Rrrats jest prostą grą kościaną typu ekstensywnego, a jej celem jest zdobycie przez gracza jak największej liczby punktów. W każdej swojej turze gracz rzuca dwiema kostkami, aż do momentu, gdy zostanie spełniony warunek stopu, bądź sam uzna, że nie chce już kontynuować. Gra kończy się w momencie, gdy z głównego stosu znikną wszystkie 31 żetonów (punktów). W grze używa się specjalnych kostek, których prawdopodobieństwo uzyskania wartości $0, 1, 2$ wynosi odpowiednio $\frac{3}{6}, \frac{2}{6}, \frac{1}{6}$.

Zasady gry są następujące:

- Gracz w swojej turze może rzucić dwoma kostkami dowolną ilość razy
- Gracz wykonuje następujące akcje w zależności od uzyskanego wyniku:
  a) Jeśli na kostce wypadło 1, gracz bierze do "ręki" żeton ze stosu głównego.
  b) Jeśli wypadło 2, gracz kradnie punkt przeciwnikowi i bierze go do swojej "ręki". Jeśli to niemożliwe gracz bierze punkt ze stosu głównego.
  c) Jeśli wypadło 0, nic się nie dzieje.
  d) Jeśli na obu kostkach wypadło 0, gracz kończy swoją turę i odkłada wszystkie punkty, jakie ma w "ręce".

  Przykład: Jeśli na kostkach wypadnie 0 i 1 gracz pobiera 1 punkt ze stosu głównego. Jeśli na kostkach wypadnie 1 i 1 gracz pobiera 2 punkty ze stosu głównego. Jeśli na kostkach wypadnie 1 i 2 gracz pobiera 1 punkt ze stosu głównego i kradnie 1 punkt przeciwnikowi.
- Po każdym rzucie gracz decyduje się na to, czy grać dalej, czy zakończyć swoją turę.
- Jeśli gracz ma więcej niż 4 punkty w "ręce" tura gracza automatycznie się kończy, a uzyskaną nadwyżkę odkłada się do stosu głównego.
- Jeśli tura dobiegła końca, to gracz przenosi wszystkie punkty uzyskane na "ręce" do swojej puli puntów osobistych.
- Gra kończy się w momencie, gdy liczba punktów na głównym stosie wyniesie 0,

W tym przypadku strategia, której będziemy szukać będzie oparta o 6 parametrów $A_1, A_2, B_1, B_2, C_1, C_2$. Przy pomocy tych paramentów wyznaczymy kolejne 6 współczynników $\alpha_1 = \exp(A_1), \alpha_2 = \exp(A_2), \beta_1 = \exp(B_1), \beta_2 = \exp(B_2), \gamma_1 = \exp(C_1), \gamma_2 = \exp(C_2)$. Naszym celem jest wyznaczenie prawdopodobieństwa zdecydowania się na dalszy rzut kośćmi w zależności od ilości punktów posiadanych na "ręce". Wprowadźmy zatem zmienną losową $Y|K = k, k = \{1, 2, 3\}$. Posłuży ona nam do wyznaczenia prawdopodobieństwa, cze gracz decyduje się na dalszy rzut kośćmi ($Y = 1$) w zależności od ilości posiadanych punktów na "ręce". Prawdopodobieństwa wprowadzonej zmiennej losowej określamy w następujący sposób:

$$\mathbb{P}(Y = 1|K = 1) = \alpha_1/(\alpha_1 + \alpha_2),$$
$$\mathbb{P}(Y = 1|K = 2) = \beta_1/(\beta_1 + \beta_2),$$
$$\mathbb{P}(Y = 1|K = 3) = \gamma_1/(\gamma_1 + \gamma_2).$$

Chapter 5

Wyniki działania algorytmów dla gier

Do prezentacji wyników działania zaproponowanych algorytmów wykorzystamy dwie tabele. Pierwsza z nich będzie przedstawiać prawdopodobieństwo wygranej strategii osiągniętej za pomocą naszych algorytmów w porównaniu z losową strategią początkową. Druga tabela natomiast przedstawi prawdopodobieństwo wygranej pomiędzy uzyskanymi algorytmami.

Wyniki dla gry Wojna

Wyniki prezentowane w Tabeli 5.1 oraz Rysunku 5.1 pokazują, że każdy z naszych algorytmów zdołał znaleźć strategię dającą lepszy wynik niż losowa strategia początkowa. Spoglądając na Tabele 5.2 widzimy, że prawdopodobieństwa wygranej dla naszych algorytmów są sobie równe, co wynika z faktu, że dla gry w Wojnę zastosowane algorytmy znalazły identyczne zwycięskie strategie. Co więcej, wyznaczona strategia może być łatwo zinterpretowana przez człowieka. W grze Wojna istnieje optymalna strategia prosta, polegająca na układaniu kart zawsze od największej do najmniejszej. Ta strategia ma 70% szans na wygraną przeciwko strategii układania kart losowo oraz 54% szans na wygraną przeciwko strategii układania kart malejąco (symulując 10000 gier). Oznacza to, że najbardziej powszechną strategią w tej grze, jest zarazem strategia silnie przegrywająca.

Wyniki dla gry Rrrats

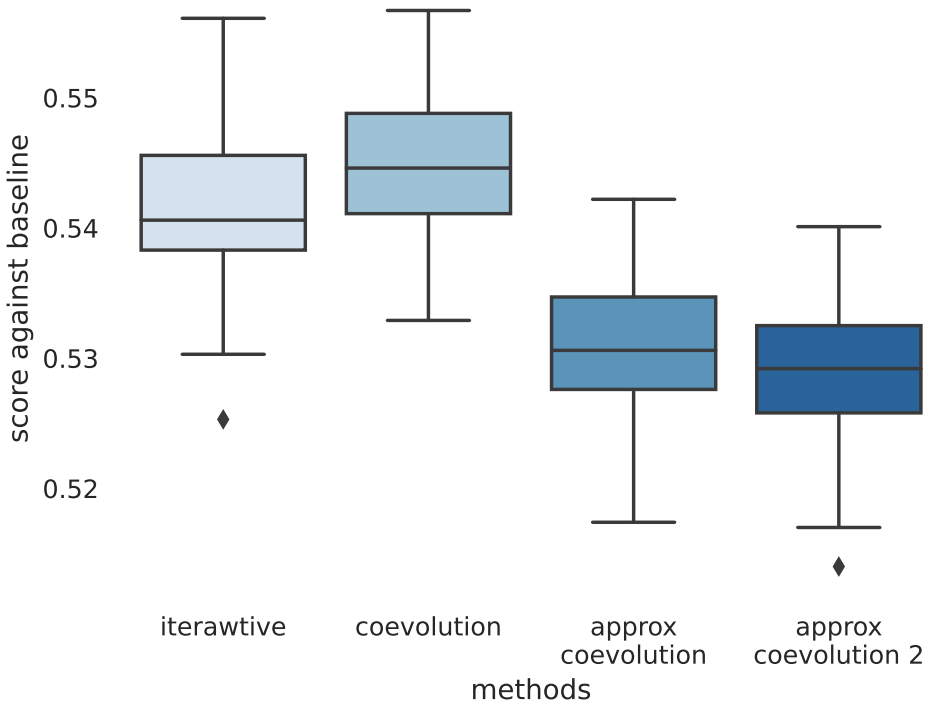
Wyniki prezentowane w Tabeli 5.3 oraz Rysunku 5.2 pokazują, że każdy z naszych algorytmów zdołał znaleźć strategię dającą lepszy wynik niż losowa strategia początkowa. Spoglądając na Tabele 5.4 widzimy, że najlepszą strategię znalazł algorytm Approximate Coevolution. Co więcej, wyznaczona strategia może być łatwo zinterpretowana przez człowieka. W grze Rrrats istnieje optymalna strategia mieszana. Strategią tą jest kończenie tury gracza z prawdopodobieństwem równym 64% jeśli gracz uzyskał 3 punkty na "ręce".

	iterawtive	coevolution	approx coevolution	approx coevolution 2
mean	0,541617	0,545225	0,530655	0,528750
std	0,005623	0,005158	0,004900	0,005244
min	0,525300	0,532900	0,517400	0,514000
25%	0,538300	0,541100	0,527600	0,525800
50%	0,540600	0,544600	0,530600	0,529200
75%	0,545575	0,548800	0,534700	0,532500
max	0,556100	0,556700	0,542200	0,540100

Table 5.1: Rezultaty uzyskanych strategii przeciwko losowej strategii początkowej dla gry Wojna.

	iterative	real coevol	approx coevol	approx coevol 2
iterative	0,501 ± 0,005	0,502 ± 0,006	0,498 ± 0,003	0,502 ± 0,005
real coevol	0,501 ± 0,005	0,498 ± 0,007	0,5 ± 0,004	0,498 ± 0,005
approx coevol	0,502 ± 0,003	0,498 ± 0,005	0,498 ± 0,006	0,498 ± 0,005
approx coevol 2	0,498 ± 0,008	0,5 ± 0,004	0,495 ± 0,004	0,498 ± 0,004

Table 5.2: Porównanie prawdopodobieństwa wygranej między strategiami uzyskanymi przez badane algorytmy dla gry Wojna. Wartość po "±" jest odchyleniem standardowym.



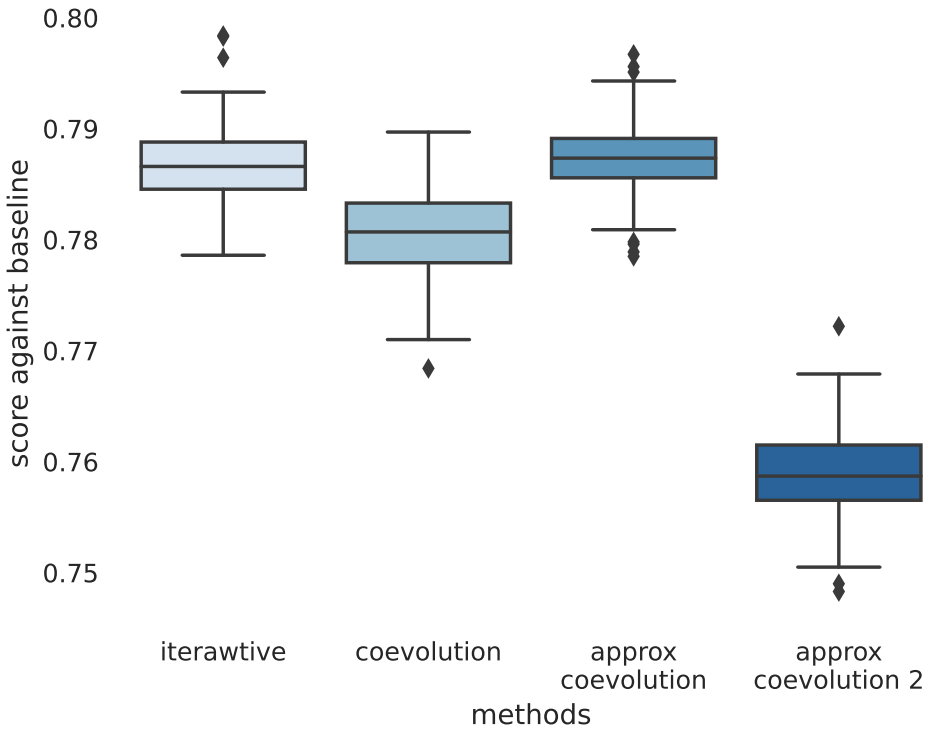
Rysunek 5.1: Wykresy pudełkowe przestawiające rezultaty uzyskanych strategii przeciwko strategii początkowej dla gry Wojna.

	iterawtive	coevolution	approx coevolution	approx coevolution 2
mean	0,786629	0,780805	0,787310	0,759073
std	0,003581	0,004265	0,003586	0,004170
min	0,778600	0,768400	0,778500	0,748300
25%	0,784550	0,777925	0,785575	0,756525
50%	0,786600	0,780700	0,787350	0,758700
75%	0,788800	0,783300	0,789125	0,761500
max	0,798400	0,789700	0,796700	0,772200

Table 5.3: Rezultaty uzyskanych strategii przeciwko losowej strategii początkowej dla gry Rrrats.

	iterative	real coevol	approx coevol	approx coevol 2
iterative	0,519 ± 0,005	0,525 ± 0,005	0,518 ± 0,006	0,564 ± 0,004
real coevol	0,508 ± 0,005	0,515 ± 0,005	0,509 ± 0,005	0,555 ± 0,005
approx coevol	0,519 ± 0,005	0,525 ± 0,005	0,519 ± 0,005	0,565 ± 0,006
approx coevol 2	0,456 ± 0,005	0,464 ± 0,005	0,457 ± 0,005	0,507 ± 0,005

Table 5.4: Porównanie prawdopodobieństwa wygranej między strategiami uzyskanymi przez badane algorytmy dla gry Rrrats. Wartość po "±" jest odchyleniem standardowym.



Rysunek 5.2: Wykresy pudełkowe przestawiające rezultaty uzyskanych strategii przeciwko losowej strategii początkowej dla gry Rrrats.

Podsumowanie

W pracy skupiliśmy się na analizie algorytmów typu LEBR, których głównym zadaniem jest szacowanie prawdopodobieństwa wygranej graczy. W celu ulepszenia ich działania zaproponowaliśmy nowe algorytmy nazwane ILEBR. Poprzez analizę nierówności Hoeffdinga i empirycznej nierówności Bernsteina, udało nam się określić maksymalną liczbę testów potrzebnych do zakończenia działania algorytmów typu ILEBR oraz wyznaczyliśmy wzór na prawdopodobieństwo popełnienia błędu dla badanych algorytmów. Ponadto opracowaliśmy metodę doboru parametrów początkowych, która pozwala ograniczyć ostateczny błąd do poziomu nieprzekraczającego żądanego prawdopodobieństwa α. Wszystkie nasze teoretyczne rozważania zostały potwierdzone przez symulacje komputerowe.

Dodatkowo zastosowaliśmy te metody do algorytmów genetycznych oraz zaproponowaliśmy nowy algorytm generacyjny. Przy ich pomocy udało nam się symulacyjnie wyznaczyć optymalną strategię w testowanych dwuosobowych grach częściowo obserwowalnych.

Dodatek

5.1 Dowód Lematu 1.14

Proof. Niech $X_1, X_2, \ldots, X_t$ będzie ciągiem i.i.d. zmiennych losowych, takim że $0 \leqslant X_i \leqslant 1$. Dodatkowo niech liczba rozegranych gier będzie funkcją zależną od k ($f(k) = t$) oraz niech $\delta_k = \frac{\delta}{g(k)}$ gdzie $\delta \geqslant \sum_{k=1}^{\infty} \frac{\delta}{g(k)}$ i $\ln(g(k)) \in o(f(k))$, wtedy

$$0 \leqslant X_i \leqslant 1 \implies \overline{\sigma}_t^2 \leqslant \frac{1}{4}. \tag{5.1}$$

Podkładając wynik (5.1) do (1.1) otrzymujemy

$$\epsilon_{f(k),k} \leqslant \sqrt{\frac{\ln(\frac{3g(k)}{\delta})}{2f(k)}} + \frac{3\ln(\frac{3g(k)}{\delta})}{f(k)}.$$

Z założeń wiemy, że $ln(g(k)) \in o(f(k))$, zatem

$$\lim_{k\to\infty} \frac{\ln(\frac{3g(k)}{\delta})}{f(k)} = 0.$$

Co ostatecznie z twierdzenia o trzech ciągach daje nam $e_{f(k),k} = 0$. □

5.2 Dowód nierówności (2.6)

Proof. Prawdopodobieństwo popełnienia błędu α jest równe sumie prawdopodobieństw pomyłki w momencie przeprowadzenia k-tego testu plus prawdopodobieństwo pomyłki po przeprowadzaniu $n_{\max}$ testów. Z założeń wiemy, że teoretyczna warowność $\mu > 0{,}5$. Zatem

$$\begin{aligned}
\alpha = {} & \mathbb{P}_\mu(\overline{X}_{f(1)} + \epsilon_1 \leqslant 0{,}5) + \\
& \mathbb{P}_\mu(\overline{X}_{f(1)} + \epsilon_1 > 0{,}5)\mathbb{P}_\mu(\overline{X}_{f(2)} + \epsilon_2 \leqslant 0{,}5) + \\
& \cdots + \\
& \mathbb{P}_\mu(\overline{X}_{f(1)} + \epsilon_1 > 0{,}5)\mathbb{P}_\mu(\overline{X}_{f(2)} + \epsilon_2 > 0{,}5) \ldots \mathbb{P}_\mu(\overline{X}_{f(n_{\max})} + \epsilon_{n_{\max}} \leqslant 0{,}5) + \\
& \mathbb{P}_\mu(\overline{X}_{f(1)} + \epsilon_1 > 0{,}5) \ldots \mathbb{P}_\mu(\overline{X}_{f(n_{\max})} + \epsilon_{n_{\max}} > 0{,}5)\mathbb{P}_\mu(\overline{X}_{f(n_{\max})} \leqslant 0{,}5).
\end{aligned}$$

Korzystając z prawdopodobieństwa zdarzenia przeciwnego otrzymujemy

$$\begin{aligned}
\alpha = {} & 1 - \mathbb{P}_\mu(\overline{X}_{f(1)} + \epsilon_1 > 0{,}5) + \\
& \mathbb{P}_\mu(\overline{X}_{f(1)} + \epsilon_1 > 0{,}5)(1 - \mathbb{P}_\mu(\overline{X}_{f(2)} + \epsilon_2 > 0{,}5)) + \\
& \cdots + \\
& \mathbb{P}_\mu(\overline{X}_{f(1)} + \epsilon_1 > 0{,}5)\mathbb{P}_\mu(\overline{X}_{f(2)} + \epsilon_2 > 0{,}5) \ldots (1 - \mathbb{P}_\mu(\overline{X}_{f(n_{\max})} + \epsilon_{n_{\max}} > 0{,}5)) + \\
& \mathbb{P}_\mu(\overline{X}_{f(1)} + \epsilon_1 > 0{,}5) \ldots \mathbb{P}_\mu(\overline{X}_{f(n_{\max})} + \epsilon_{n_{\max}} > 0{,}5)\mathbb{P}_\mu(\overline{X}_{f(n_{\max})} \leqslant 0{,}5).
\end{aligned}$$

$$\begin{aligned}\alpha =& 1 - \prod_{k=1}^{n_{\max}} \mathbb{P}_\mu(\overline{X}_{f(k)} + \epsilon_k > 0{,}5) + \mathbb{P}_\mu(\overline{X}_{f(n_{\max})} \leqslant 0{,}5) \prod_{k=1}^{n_{\max}} \mathbb{P}_\mu(\overline{X}_{f(k)} + \epsilon_k > 0{,}5) \\ =& 1 - (1 - \mathbb{P}_\mu(\overline{X}_{f(n_{\max})} \leqslant 0{,}5)) \prod_{k=1}^{n_{\max}} \mathbb{P}_\mu(\overline{X}_{f(k)} + \epsilon_k > 0{,}5) \\ =& 1 - \mathbb{P}_\mu(\overline{X}_{f(n_{\max})} > 0{,}5) \prod_{k=1}^{n_{\max}} \mathbb{P}_\mu(\overline{X}_{f(k)} + \epsilon_k > 0{,}5).\end{aligned}$$

□

5.3 Dowód wyrażenia (2.19)

Proof. Chcemy znaleźć minimalne $n_{\max}$, dla którego nasz błąd będzie nie większy niż żądane α. Zatem takie $n_{\max}$ będzie spełniać zależność

$$n_{\max} = \min\left\{n \in \mathbb{N}^+ : F_\mu(n) < \alpha\right\}. \tag{5.2}$$

Podstawiając (5.2) do 1.11 otrzymujemy, że dla $\delta_k = \frac{\delta}{n_{\max}}$ i $t = f(n_{\max})$

$$\epsilon \leqslant \sqrt{\frac{\ln(2\frac{n_{\max}}{\delta})}{2f(n_{\max})}} \implies \delta \leqslant \frac{2n_{\max}}{\exp(2\epsilon^2 f(n_{\max}))}$$

□

Bibliography

[1] Cauwet, M.-L., Teytaud, O. Surprising strategies obtained by stochastic optimization in partially observable games. In *2018 IEEE Congress on Evolutionary Computation (CEC)* (2018), IEEE, pp. 1–8.

[2] Collet, P., Lutton, E., Raynal, F., Schoenauer, M. Polar IFS+Parisian Genetic Programming=Efficient IFS Inverse Problem Solving. *Genetic Programming and Evolvable Machines 1*, 4 (2000), 339–361.

[3] Droste, S., Jansen, T., Wegener, I. A rigorous complexity analysis of the (1+1) evolutionary algorithm for separable functions with boolean inputs. *Evolutionary Computation 6*, 2 (1998), 185–196.

[4] Figielska, E. Algorytmy ewolucyjne i ich zastosowania. 81–92.

[5] Heidrich-Meisner, V., Igel, C. Non-linearly increasing resampling in racing algorithms. In *European Symposium on Artificial Neural Networks* (2011), Evere, Belgium: d-side publications, pp. 465–470.

[6] Maron, O., Moore, A. Hoeffding races: Accelerating model selection search for classification and function approximation. In *Proceedings of (NeurIPS) Neural Information Processing Systems* (November 1993), Morgan Kaufmann, pp. 59 – 66.

[7] Mnih, V., Szepesvári, C., Audibert, J.-Y. Empirical bernstein stopping. In *Proceedings of the 25th international conference on Machine learning - ICML '08* (2008), ACM Press.

[8] Płatkowski, T. Wstęp do teorii gier. *Uniwersytet Warszawski* (2012).

[9] Prisner, E. *Game theory through examples*, vol. 46. American Mathematical Soc., 2014.

[10] Stengel, B. V. Chapter 45 computing equilibria for two-person games. In *Handbook of Game Theory with Economic Applications*. Elsevier, 2002, pp. 1723–1759.